Topic: Data Storage

Goals: By the end of this topic, we will discuss…

- *what happens when functions are stored in memory?*

Acknowledgements: These class notes build on the content of my previous courses as well as the work of R. Jordan Crouser, and Jeffrey S. Castrucci.

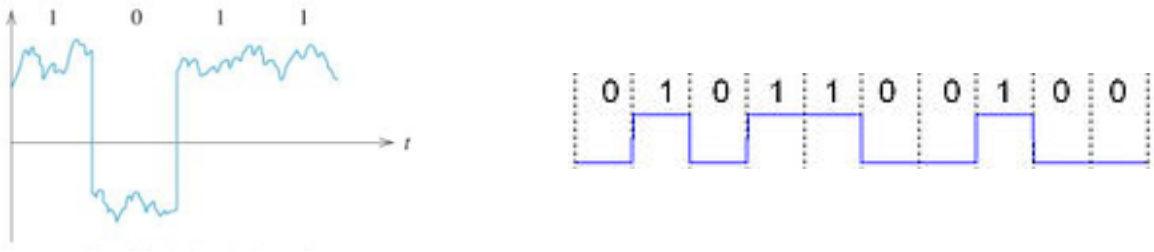
Data

Let's talk about stacks..... as in stackoverflow but first data storage.

Thinking Machines

~1837 (Ada Lovelace and Charles Babbage) started thinking about how they might go about designing a “thinking machine” (known as the Analytical Engine), they realized that any such machine would need to be able to perform 4 basic tasks: input, output, process, and storage. Those early computers were made out of wood and metal, and you literally turned a crank to make the “computation” happen.

Data Storage



Idea 1: Turning analog electrical signals into digital signals (bits).
As we add more bits, the amount of information we can store grows exponentially.

- 1 bit: OFF ON
- 2 bits: OFF OFF | OFF ON | ON OFF | ON ON
- 3 bits:
- 4 bits:

In base-10, each place is a power of 10, and each digit can take on a value from 0 to 9.
Example: 12 (Twelve) 1 "ten" and 2 "ones" or $(1*10^1) + (2*10^0) = 12$

Using bits to represent numbers:
In base-2 (“binary”), each place represents a power of 2, and each digit can take on a value of either 0 or 1: 1100_2

$(1*2^3) + (1*2^2) + (0*2^1) + (0*2^0) = 12_{10}$

$(1*8) + (1*4) + (0*2) + (0*1) = 12$

How much can we represent?

- 8 bites -> we can represent the numbers 0 to 255 (8 bits is called a “byte”)
- 32 bits -> we can represent numbers ***> 4 billion***
- 266 bits -> we can represent ***more unique numbers*** than there are believed to be ***atoms in the universe***

Generally the more information required the more the storage. For example:

iPhone camera (8 megapixels): 3296 x 2472 pixels
each requires 4 bytes to represent RGB + opacity
32,590,848 bits ≈ **4MB** (1MB = 1024 bytes, 1 byte = 8 bits)

HD video: 1920 x 1080 pixels, 30fps
5 minutes of video = 300 seconds = 9000 frames
2,073,600 bits per frame ≈ **2.33GB** (1GB = 1024MB)

Your program data is stored in memory (RAM -> random-access memory).

Two forms of memory: Stack and Heap.

The Stack

- stores program call (including function calls
- operates as last in, first out (LIFO or FILO) queue
- limited space -> stack overflow errors also recursive depth errors

the program

```
def f3(a):
    if (a == 1):
      return 1
    else:
      x = f3(a-1)+1

f3(100)
```

in memory

The Heap

- heap for dynamic memory allocation, but storage is not ordered
- heap size is only limited by the size of virtual memory
- program variables stored in heap